A Path Forward to ~~Intrusive~~ Embedded Sensitivity Analysis, Uncertainty Quantification and Optimization

Eric Phipps

Optimization and Uncertainty Quantification Department

Sandia National Laboratories

Albuquerque, NM USA

etphipp@sandia.gov

NEAMS VU Workshop

April 7-8, 2009

The Challenge For Embedded Methods

- **Embedded/Intrusive Methods:**
  - Exploiting simulation code structure for improved performance (speed, accuracy, robustness,…)
  - Requiring more information from code beyond repeated simulation
- **Performance advantages often remarkable**
  - Intrusiveness into code often also significant
- **Bridging the gap between algorithms research and applications is the challenge**
  - Requires significant effort and foresight of code developers
  - A priori unclear which, if any, methods will significantly impact application
- **A path forward is necessary that**
  - Enables a wide variety of important embedded methods
  - Eases burden on simulation code developers

Overview

- **Sensitivity Analysis**
  - **Forward & Adjoint methods**
- **Uncertainty Quantification**
  - **Stochastic Galerkin**
  - **Adjoint**
- **Optimization**
  - **NAND to SAND**
- **A path forward**
  - **Code interfaces**
  - **Automatic Differentiation**

Mathematical Model

$$
\begin{aligned}
0 &= f(\dot{u}(t), u(t), p, t), \quad t \in [t_0, t_f] \\
u(t_0) &= u_0(p) \\
\dot{u}(t_0) &= \dot{u}_0(p) \\
v(p) &= \int_{t_0}^{t_f} g(\dot{u}(t), u(t), p, t)dt + h(\dot{u}(t_f), u(t_f), p)
\end{aligned}
$$

Steady-State Embedded Sensitivity Analysis

$$f(u,p) = 0, \quad v(p) = h(u,p)$$

Forward sensitivities

$$\frac{\partial v}{\partial p} = \frac{\partial h}{\partial u}\left(-\frac{\partial f}{\partial u}^{-1}\frac{\partial f}{\partial p}\right) + \frac{\partial h}{\partial p}$$

- **Cost scales with number of *parameters***
- **Solve system Jacobian**

Adjoint sensitivities

$$\frac{\partial v}{\partial p}^T = \frac{\partial f}{\partial p}^T\left(-\frac{\partial f}{\partial u}^{-T}\frac{\partial h}{\partial u}^T\right) + \frac{\partial h}{\partial p}^T$$

- **Cost scales with number of *observation functions***
- **Solve system Jacobian-transpose**

- **Small extension for Newton-based codes**
- **Sensitivity (linear) solves significantly cheaper than (nonlinear) state solves**
- **Accurate derivatives critical (can't use approximate Jacobian)**
- ***Simulation code* must evaluate observation functions & gradients**

Transient Embedded Sensitivity Analysis

Forward sensitivities

$$\frac{\partial f}{\partial \dot{u}}\left(\frac{\partial \dot{u}}{\partial p}\right)+\frac{\partial f}{\partial u}\left(\frac{\partial u}{\partial p}\right)+\frac{\partial f}{\partial p}=0, \quad t\in[t_0,t_f],$$

$$\frac{\partial u}{\partial p}(t_0)=\frac{\partial u_0}{\partial p}, \quad \frac{\partial \dot{u}}{\partial p}(t_0)=\frac{\partial \dot{u}_0}{\partial p},$$

$$\frac{\partial v}{\partial p}=\int_{t_0}^{t_f}\left(\frac{\partial g}{\partial \dot{u}}\frac{\partial \dot{u}}{\partial p}+\frac{\partial g}{\partial u}\frac{\partial u}{\partial p}+\frac{\partial g}{\partial p}\right)dt+$$

$$\left.\left(\frac{\partial h}{\partial \dot{u}}\frac{\partial \dot{u}}{\partial p}+\frac{\partial h}{\partial u}\frac{\partial u}{\partial p}+\frac{\partial h}{\partial p}\right)\right|_{t=t_f}$$

- **Linear ODE for sensitivities solved alongside original model**
- **Cost scales with number of *parameters***
- **Hindmarsh *et al***

Adjoint sensitivities

$$\frac{d}{dt}\left(\frac{\partial f}{\partial \dot{u}}^T\Lambda\right)-\frac{\partial f}{\partial u}^T\Lambda+\frac{\partial g}{\partial u}^T=0, \quad t\in[t_0,t_f],$$

$$\left.\left(\frac{\partial f}{\partial \dot{u}}^T\Lambda\right)\right|_{t=t_f}=\left.\frac{\partial h}{\partial u}^T\right|_{t=t_f},$$

$$\frac{\partial v}{\partial p}^T=\int_{t_0}^{t_f}\left(\frac{\partial g}{\partial p}^T-\frac{\partial f}{\partial p}^T\Lambda\right)dt+\left.\frac{\partial h}{\partial p}^T\right|_{t=t_f}+$$

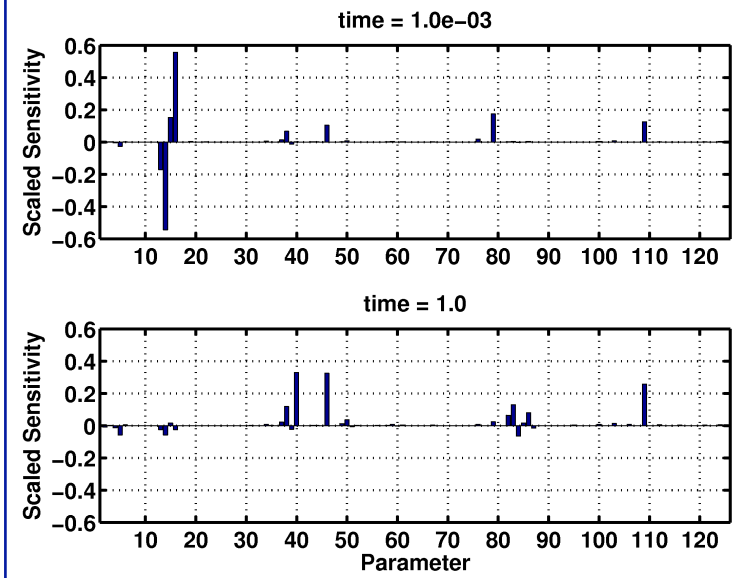
$$\frac{\partial u_0}{\partial p}^T\left.\left(\frac{\partial f}{\partial \dot{u}}^T\Lambda\right)\right|_{t=t_0}$$

- **Linear ODE for adjoint that must be integrated backward in time**
- **Requires full forward model integration first (or check-pointing)**
- **Cost scales with number of *objective functions***
- **Petzold *et al***

Costs and Benefits for Embedded SA

- **Costs & Limitations**
  - Only local analysis
  - Requires accurate derivatives
  - Adjoint approach requires specialized time integration tools
    - SUNDIALS, Trilinos/Rythmos
- **Benefits**
  - Orders-of-magnitude cheaper than global analysis
  - More accurate, efficient, and robust than finite-difference-based analysis
  - Adjoint cost independent of number of parameters
  - Foundation for optimization, error estimation, and UQ

126 sensitivities in 9 simulation's time!



Forward transient sensitivity analysis of a Charon simulation of a radiation-damaged transistor with respect to damage mechanisms using Rythmos & Sacado (Phipps et al).

Sandia National Laboratories

Embedded Stochastic Galerkin Uncertainty Quantification Methods

- **Steady-state stochastic problem:**

$$\text{Find } u(\xi) \text{ such that } f(u,\xi)=0,\ \xi:\Omega\to\Gamma\subset R^M,\ \text{density } \rho$$

- **Stochastic Galerkin method (Ghanem, …):**

$$\hat{u}(\xi)=\sum_{i=0}^{N}u_i\psi_i(\xi)\to f_i(u_0,\dots,u_N)=\int_\Gamma f(\hat{u}(y),y)\psi_i(y)\rho(y)dy=0,\quad i=0,\dots,N$$

- **Basis polynomials are tensor products of 1-D orthogonal polynomials of degree P**
  - **Gaussian (Hermite polynomials), Uniform (Legendre), …**
  - **Assumes independence of random parameters**

- **Method generates new coupled spatial-stochastic nonlinear problem**

$$0=\bar{f}(\bar{u})=\begin{bmatrix}f_0\\ f_1\\ \vdots\\ f_N\end{bmatrix},\quad \bar{u}=\begin{bmatrix}u_0\\ u_1\\ \vdots\\ u_N\end{bmatrix}$$

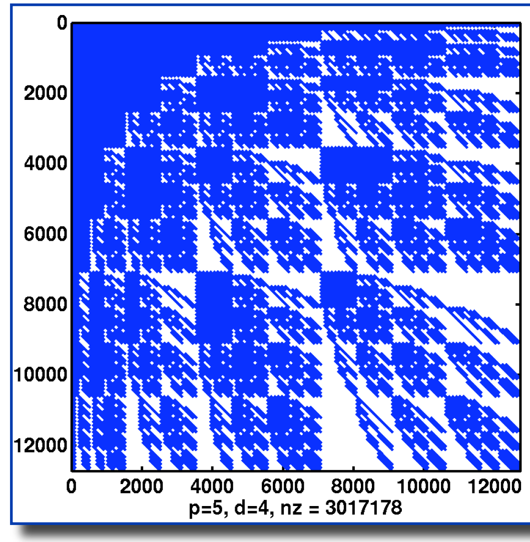
- **Total size grows rapidly with degree or dimension**
  - **Exponential convergence in degree**

$$N=\frac{(M+P)!}{M!P!}$$

Stochastic dimension M	Polynomial degree P	Number of terms N
5	3	56
	5	252
10	3	286
	5	3003
20	3	1,771
	5	~53,000
100	3	~177,000
	5	~96,000,000

Costs and Benefits of Embedded SG

- Costs & Limitations
  - R&D needed for effective implementation
    - Automated code transformation
    - Data structures and interfaces
    - Solver algorithms
  - Effectiveness in hard problems unknown
  - Likely requires significant HPC resources
  - Breaks down in presence of discontinuities



- Benefits
  - AD, quadrature and solver tools under development
    - Trilinos/Stokhos/Sacado
  - Potential for significant savings over non-intrusive methods
  - Potential for *a posteriori* error estimates
  - Generates a response surface that can be quickly sampled for
    - Probabilities, sensitivities, Bayesian methods (Marzouk *et al*)
  - Extensions
    - Local bases (Le Maitre *et al*), non-independent parameters (Wan *et al*), stochastic model reduction (Doostan *et al*)

Sandia National Laboratories

Adjoint-Based Embedded UQ Methods

- **Piecewise 1st order response surface over a grid (Estep, *et al*)**

$$f(u_0, p_0) = 0, \quad v_0 = h(u_0), \quad \left(\frac{\partial f}{\partial u}(u_0, p_0)\right)^T \Lambda = \frac{\partial h}{\partial u}(u_0)^T$$

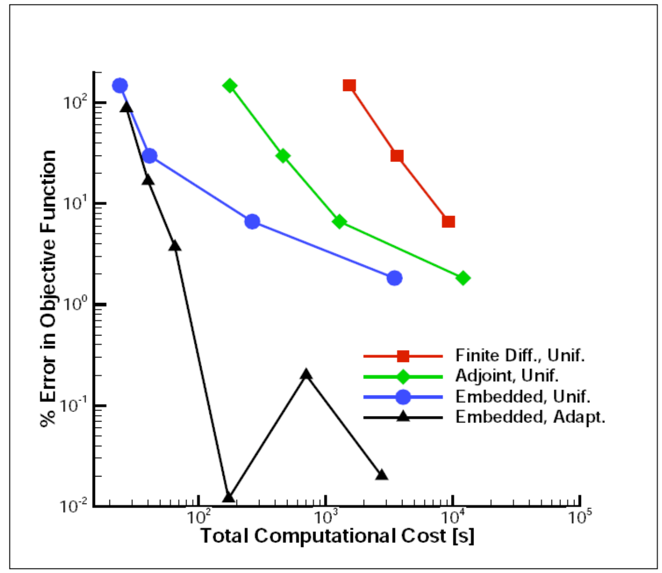
$$v(p) \approx v(p_0) - \left(\frac{\partial f}{\partial p}(u_0, p_0)(p - p_0)\right)^T \Lambda$$

- **Leverages adjoint sensitivity tools**
- **Good performance in small dimensions against Monte Carlo**
  - 1-2 orders of magnitude reduction in number of samples/grid points
  - Computing each local response surface is fast
  - Number of grid points grows exponentially in number of dimensions
  - Unknown how it compares to other UQ approaches
- **Naturally adaptive**
  - *A posteriori* error estimates and adaptivity
  - No trouble with bifurcations/discontinuities
- **Extension for inverse uncertainty problems (Butler & Estep)**
- **No general purpose tools available**

Embedded Optimization

$$\min_p h(u,p) \;\; \text{s.t.} \;\; f(u,p) = 0$$

- Optimization for
  - Model Calibration
  - Validation (computing probability models for inputs of multiscale/fidelity models, e.g., Arnst & Ghanem)
- Nested Analysis And Design (non-intrusive to semi-embedded)
  - Nonlinearly eliminate constraints
  - Compute reduced sensitivities using finite differences or embedded sensitivity techniques
  - Linear convergence
  - Small to medium parameter spaces O(1-100)
- Simultaneous Analysis and Design (embedded)
  - Solve optimization and constraints simultaneously
    - Eliminates constraint solves away from optimum
  - Built on the same tools as embedded sensitivities
  - Super-linear to quadratic convergence
  - First to second derivatives
  - Scalable to very large parameter spaces
  - Orders-of-magnitude more efficient than NAND
- R&D necessary for challenging problems
  - Globalizations
  - Non-smooth systems
  - KKT solvers for 2nd-derivative-based methods



Reduced-space (super-linear SAND) optimization of flow and transport using Trilinos/ MOOCHO. Courtesy of B. van Bloemen Waanders, SNL.

Sandia National Laboratories

A Path Forward

- Significant R&D is needed for embedded methods to impact your applications
- Application codes need to be “born” with these technologies
  - Retrofitting is difficult and almost never happens
- With the right hooks, this is feasible
  - High-level application code interfaces
    - Residuals, Jacobians, objective/observation functions, parameter deriv’s, …
  - Automatic differentiation
    - Tools to implement those interfaces

High-Level Application Code Interfaces

- Requirements for many embedded algorithms are simple
  - Set state values (u, du/dt)
  - Set parameter values (p)
  - Compute application residual (f)
  - Compute observation/objective functions (g, h)
  - Compute derivatives (df/du, df/dp, …)

- Trilinos provides a unified application interface for all of its embedded algorithms
  - Thyra::ModelEvaluator
  - Can provide decorators/wrappers for
    - SG residuals/Jacobians
    - Reduced sensitivities
    - Integration with Dakota

- Computing derivatives is usually the difficult part

Automatic Differentiation Provides Tools for Implementing Embedded Algorithm Interfaces

- Derivatives are critical for many embedded algorithms
  - Must be accurate and efficient
- Automatic differentiation provides analytic derivatives with minimal code development/maintenance
  - Derivatives at operation-level known, combined with Chain Rule
  - Any kind of first or higher-order derivative
  - SG polynomials, intervals, …
  - Automatically verified to be correct
- Good tools exist
  - Fortran -- Source transformation -- OpenAD/ADIFOR
  - C++ -- Operator overloading, templating -- Trilinos/Sacado
  - Demonstrated effectiveness, efficiency, and scalability for large-scale simulations
- Prescription for applying AD simple
  - Separate parts of the code to be differentiated from others (e.g., element residual fill) with well-defined interfaces
  - Fortran – apply source transformation to those parts
  - C++ – template those parts for operator overloading

Sandia National Laboratories

Concluding Remarks

- Potentially tremendous computational cost savings with embedded methods
- Significant algorithms R&D is necessary to realize those savings in applications
- Codes must be “born” with these technologies to reap their benefits
  - High-level application code interfaces
  - Automatic differentiation to implement those interfaces
    - Separate out differentiable pieces
    - Template those pieces (for C++ applications)
- Ideas are complementary to Dakota

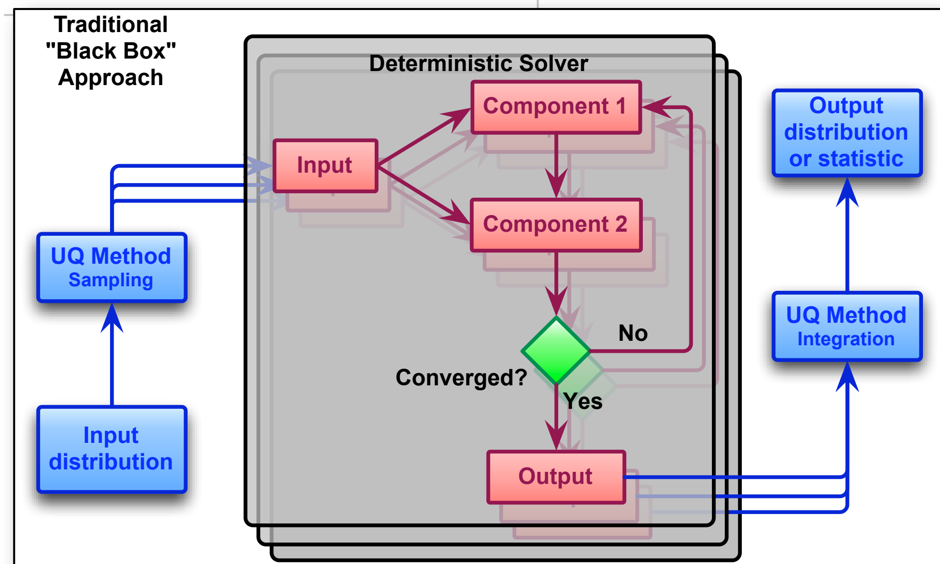
References

- Sensitivity Analysis
  - A. Hindmarsh,, P. Brown, K. Grant, S. Lee, R. Serban, D. Shumaker, and C. Woodward. "Sundials: Suite of nonlinear and differential/algebraic equation solvers." *ACM Trans. Math. Softw*. 31(3): 363–396, 2005.
  - E. Phipps, R. Bartlett, D. Gay, and R. Hoekstra. "Large-Scale Transient Sensitivity Analysis of a Radiation-Damaged Bipolar Junction Transistor via AD." *Advances in Automatic Differentiation*, C. Bischof, M. Bucker, P. Hovland, U. Naumann, and J. Utke, eds., *Lecture Notes in Computational Science and Engineering*, 2008.
- Uncertainty Quantification
  - B. Debusschere, H. Najm, P. Pebay, O. Knio, R. Ghanem, and O. L. Maitre. "Numerical challenges in the use of polynomial chaos representations for stochastic processes." *SIAM J Sci Comput*, 26(2): 698–719, 2004.
  - D. Estep and D. Neckels. "Fast and reliable methods for determining the evolution of uncertain parameters in differential equations." *Journal of Computational Physics*, 213: 530–556, 2005.
  - H. Matthies and A. Keese. "Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations." *Comput. Methods Appl. Mech. Engrg.* 194: 1295–1331, 2005.
- Optimization
  - B. van Bloemen Waanders, R. Bartlett, K. Long, P. Boggs, and A. Salinger. "Large-Scale Non-Linear Programming for PDE Constrained Optimization." Technical Report SAND2002-3198, Sandia National Laboratories, October, 2002.
- Software
  - Trilinos (Rythmos, MOOCHO, Sacado, Stokhos, …): http://trilinos.sandia.gov
  - OpenAD: http://www.mcs.anl.gov/OpenAD/, http://www.autodiff.org
  - SUNDIALS: https://computation.llnl.gov/casc/sundials/main.html

Sandia National Laboratories

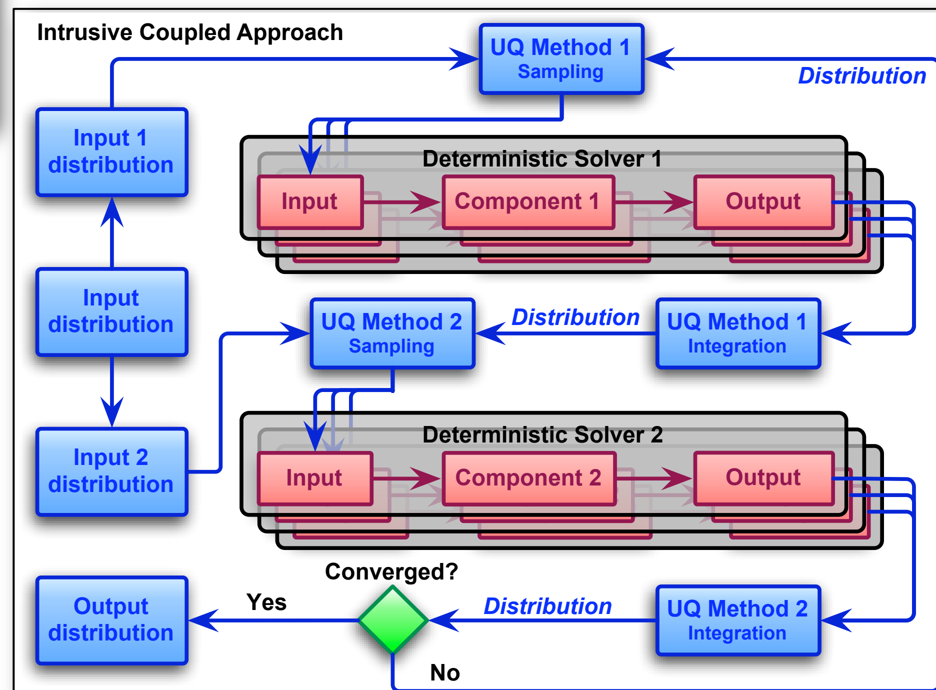
Auxiliary Slides

Coupled System Embedded UQ Research



- Invert layering of UQ around system simulation
  - Apply UQ to each component separately
  - Stochastic coupled solver technology
- Potentially orders of magnitude savings
  - Heterogeneous UQ
  - Stochastic dimension reduction

- Coupled systems generate large dimensional stochastic spaces
  - 10 for component 1 + 10 for component 2 = 20 dimensions
  - Cost grows rapidly with dimension
- Inverted approach breaks growth
  - 1-dimensional interface between components
  - 2 11-dimensional UQ problems



What is Automatic Differentiation (AD)?

- **Technique to compute analytic derivatives without hand-coding the derivative computation**
- **How does it work -- freshman calculus**
  - **Computations are composition of simple operations (+, *, sin(), etc…) with known derivatives**
  - **Derivatives computed line-by-line, combined via chain rule**
- **Derivatives accurate as original computation**
  - **No finite-difference truncation errors**
- **Provides analytic derivatives without the time and effort of hand-coding them**

$$y = \sin(e^x + x \log x), \quad x = 2$$

		x	$\frac{d}{dx}$
$x \leftarrow 2$	$\frac{dx}{dx} \leftarrow 1$	2.000	1.000
$t \leftarrow e^x$	$\frac{dt}{dx} \leftarrow t\frac{dx}{dx}$	7.389	7.389
$u \leftarrow \log x$	$\frac{du}{dx} \leftarrow \frac{1}{x}\frac{dx}{dx}$	0.301	0.500
$v \leftarrow xu$	$\frac{dv}{dx} \leftarrow u\frac{dx}{dx} + x\frac{du}{dx}$	0.602	1.301
$w \leftarrow t + v$	$\frac{dw}{dx} \leftarrow \frac{dt}{dx} + \frac{dv}{dx}$	7.991	8.690
$y \leftarrow \sin w$	$\frac{dy}{dx} \leftarrow \cos(w)\frac{dw}{dx}$	0.991	-1.188

AD Takes Three Basic Forms

$$x \in \mathrm{R}^n,\ f : \mathrm{R}^n \to \mathrm{R}^m$$

- **Forward Mode:**

$$(x,\ V) \longrightarrow \left(f,\ \frac{\partial f}{\partial x}V\right)$$

  - **Propagate derivatives of intermediate variables w.r.t. independent variables forward**
  - **Directional derivatives, tangent vectors, square Jacobians,** $\partial f/\partial x$ **when** $m \geq n$

- **Reverse Mode:**

$$(x,\ W) \longrightarrow \left(f,\ W^T\frac{\partial f}{\partial x}\right)$$

  - **Propagate derivatives of dependent variables w.r.t. intermediate variables backwards**
  - **Gradient of a scalar value function with complexity** ≈ 4 ops(f)
  - **Gradients, Jacobian-transpose products (adjoints),** $\partial f/\partial x$ **when** $n > m$

- **Taylor polynomial mode:**

$$x(t) = \sum_{k=0}^{d} x_k t^k \longrightarrow \sum_{k=0}^{d} f_k t^k = f(x(t)) + O(t^{d+1}), \quad f_k = \frac{1}{k!}\frac{d^k}{dt^k}f(x(t))$$

- **Basic modes combined for higher derivatives:**

$$\frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}V_1\right)V_2,\quad W^T\frac{\partial^2 f}{\partial x^2}V,\quad \frac{\partial f_k}{\partial x_0}$$

Our AD Research is Distinguished by Tools & Approach for Large-Scale Codes

- **Many AD tools and research projects**
  - Most geared towards Fortran (ADIFOR, OpenAD)
  - Most C++ tools are slow (ADOL-C)
  - Most applied in black-box fashion

- **Sacado: Operator overloading AD tools for C++ applications**
  - ✓ Multiple highly-optimized AD data types
  - ✓ Transform to template code & instantiate on Sacado AD types
  - ✓ Apply AD only at the "element level"

- **This is the only successful, sustainable approach for large-scale C++ codes!**

- **Directly impacting QASPR through Charon**
  - ✓ Analytic Jacobians and parameter derivatives

Trilinos

QASPR
QUALIFICATION ALTERNATIVES TO SPR

CHARON

Basic Sacado C++ Example

```cpp
#include "Sacado.hpp"

// The function to differentiate
template <typename ScalarT>
ScalarT func(const ScalarT& a, const ScalarT& b, const ScalarT& c) {
  ScalarT r = c*std::log(b+1.)/std::sin(a);

  return r;
}

int main(int argc, char **argv) {
  double a = std::atan(1.0);                              // pi/4
  double b = 2.0;
  double c = 3.0;
  int num_deriv = 2;                                      // Number of independent variables

  // Fad objects
  Sacado::Fad::DFad<double> afad(num_deriv, 0, a); // First (0) indep. var
  Sacado::Fad::DFad<double> bfad(num_deriv, 1, b); // Second (1) indep. var
  Sacado::Fad::DFad<double> cfad(c);               // Passive variable
  Sacado::Fad::DFad<double> rfad;                  // Result

  // Compute function
  double r = func(a, b, c);

  // Compute function and derivative with AD
  rfad = func(afad, bfad, cfad);

  // Extract value and derivatives
  double r_ad = rfad.val();     // r
  double drda_ad = rfad.dx(0);  // dr/da
  double drdb_ad = rfad.dx(1);  // dr/db
```

Efficiency of AD in Charon

Set of N hypothetical chemical species:

$$2X_j \rightleftharpoons X_{j-1} + X_{j+1}, \quad j = 2, \ldots, N-1$$

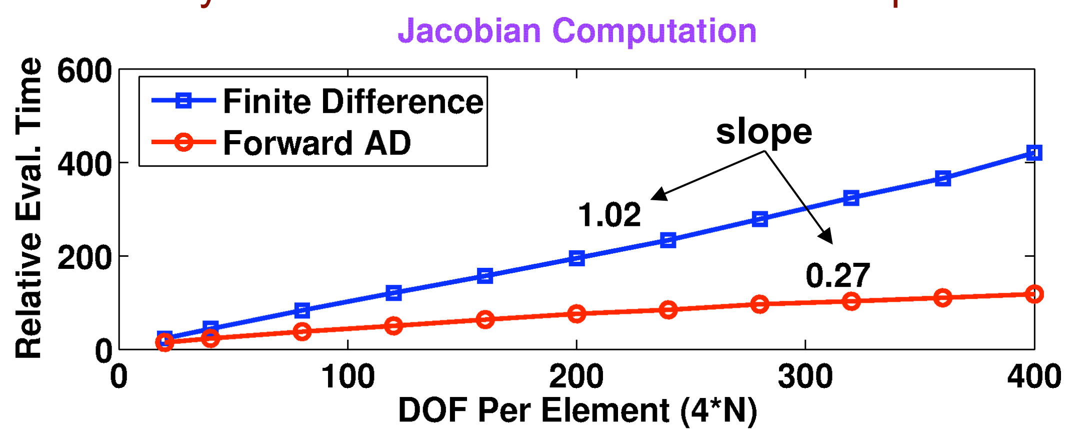
Steady-state mass transfer equations:

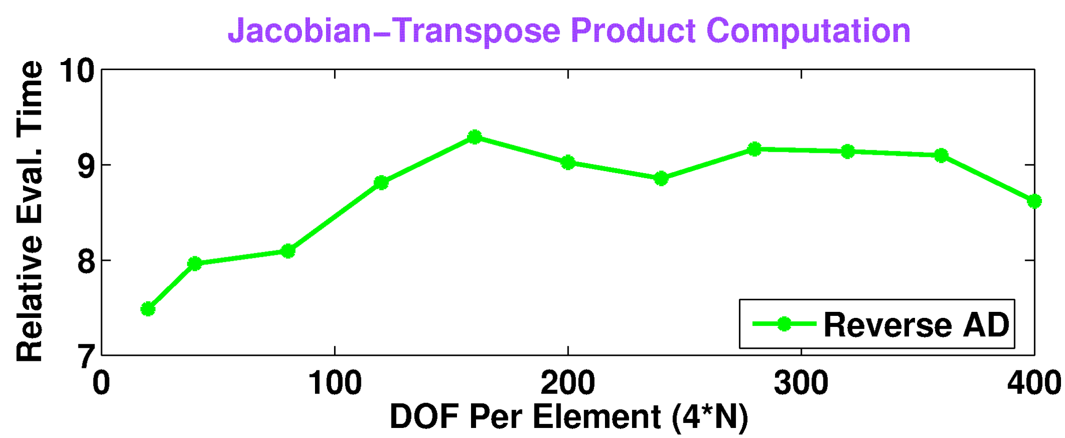
$$\nabla^2 Y_j + \mathrm{u} \cdot \nabla Y_j = \dot{\omega}_j, \quad j = 1, \ldots, N-1$$

$$\sum_{j=1}^{N} Y_j = 1$$

- **Forward mode AD**
  - **Faster than FD**
  - **Better scalability in number of PDEs**
  - **Analytic derivative**
  - **Provides Jacobian for all Charon physics**
- **Reverse mode AD**
  - **Scalable adjoint/gradient** $J^T w = \nabla(w^T f(x))$

Efficiency of the element-level derivative computation

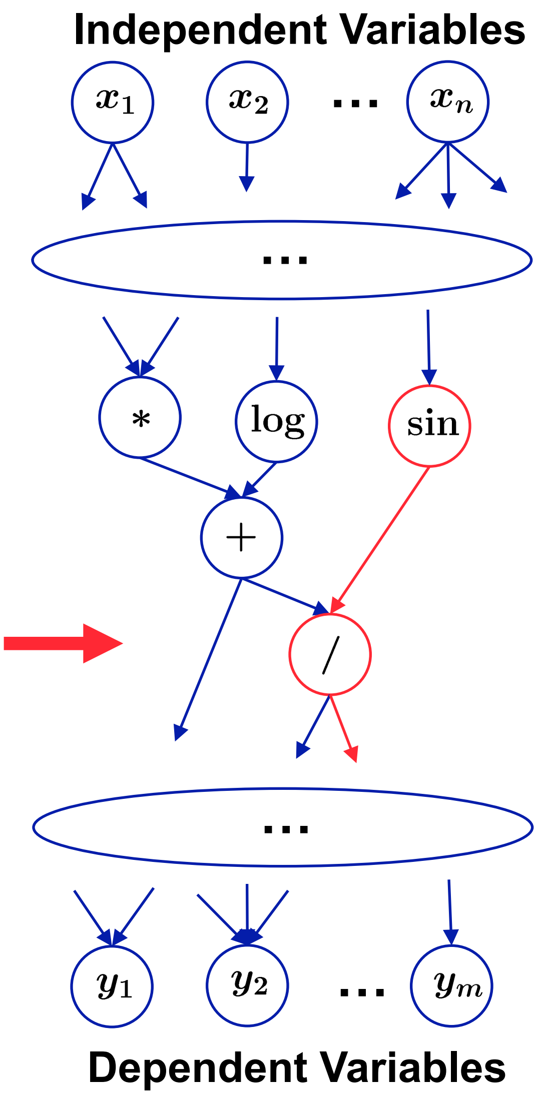




Sandia National Laboratories

Verification of Automatic Differentiation

- Verification of the AD tools
  - Unit-test with respect to known derivatives
  - Composite tests
    - Compare to other tools
    - Compare to hand-derived
    - Compare to finite differences
- Verification of AD in application code
  - Compiler drastically simplifies this
  - All of the standard hand-coded verification techniques
    - Compare to finite differences
    - Nonlinear convergence



Compiler type mechanism will not allow breaking the chain from independent to dependent variables

Charon Drift-Diffusion Formulation with Defects

CHARON

QASPR
QUALIFICATION ALTERNATIVES TO SPR

Current Conservation for e- and h+

$$\frac{\partial n}{\partial t} - \nabla \cdot J_n = -R_n(\psi, n, p, Y_1, \ldots, Y_N), \quad J_n = -n\mu_n \nabla\psi + D_n \nabla n$$

$$\frac{\partial p}{\partial t} + \nabla \cdot J_p = -R_p(\psi, n, p, Y_1, \ldots, Y_N), \quad J_p = -p\mu_p \nabla\psi - D_p \nabla p$$

Defect Continuity

$$\frac{\partial Y_i}{\partial t} + \nabla \cdot J_{Y_i} = -R_{Y_i}(\psi, n, p, Y_1, \ldots, Y_N), \quad J_{Y_i} = -\mu_i Y_i \nabla\psi - D_i \nabla Y_i$$

Electric potential

$$-\nabla(\varepsilon \nabla \psi(x)) = -q\left(p(x) - n(x) + N_D^+(x) - N_A^-(x)\right) - \sum_{i=1}^{N} q_i Y_i(x)$$

Recombination/ generation source terms R_X

Include electron capture and hole capture by defect species and reactions between various defect species

Electron emission/ capture

$Z^i \leftrightarrow Z^{i+1} + e^-$

Activation Energy

$$R_{[Z^i \to Z^{i+1} + e^-]} \propto \sigma_{[Z^i \to Z^{i+1} + e^-]} Z^i \exp\left(\frac{\Delta E_{[Z^i \to Z^{i+1} + e^-]}}{kT}\right)$$

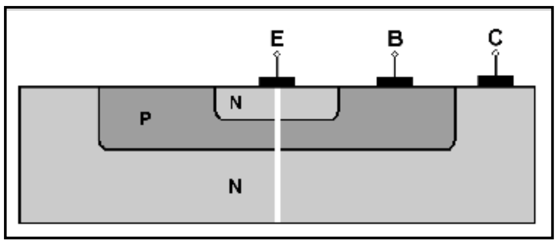
Cross section

Sandia National Laboratories

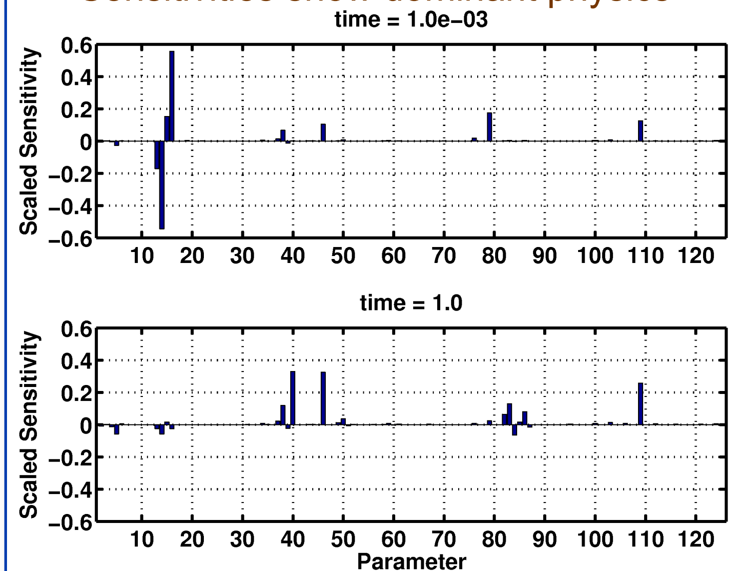
Rythmos Sensitivity Analysis Capability Demonstrated on the QASPR Simple Prototype*

*Phipps *et al*

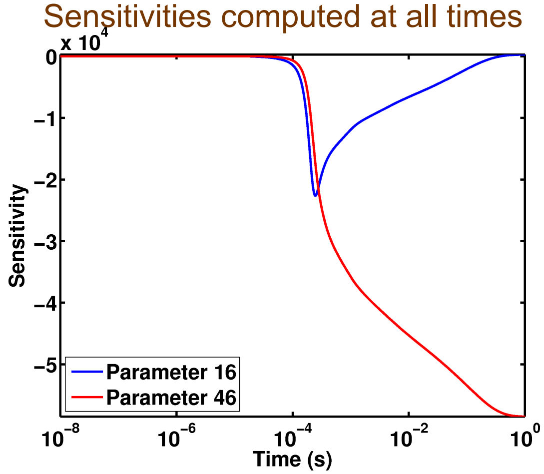
- **Bipolar Junction Transistor**
- **Pseudo 1D strip (9x0.1 micron)**
- **Full defect physics**
- **126 parameters**



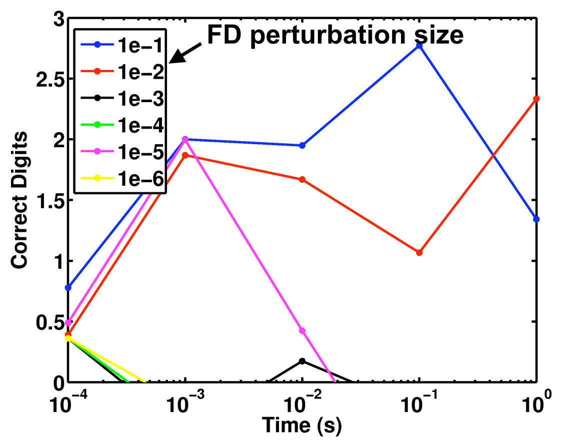
Sensitivities show dominant physics



Sensitivities computed at all times



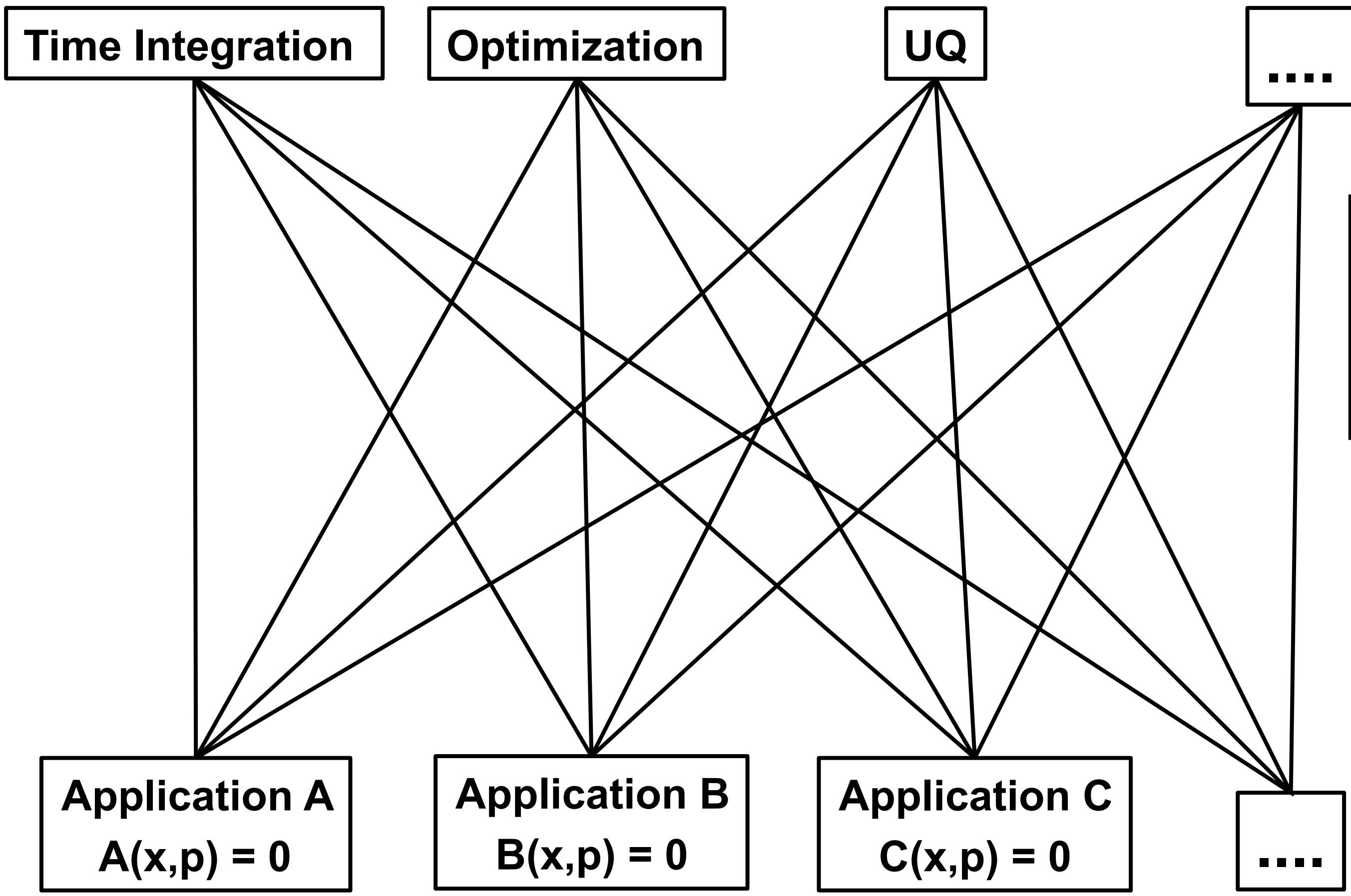
1st-order Finite Difference Accuracy



Comparison to FD:

- ✓ **Sensitivities at all time points**
- ✓ **More accurate**
- ✓ **More robust**
- ✓ **14x faster!**

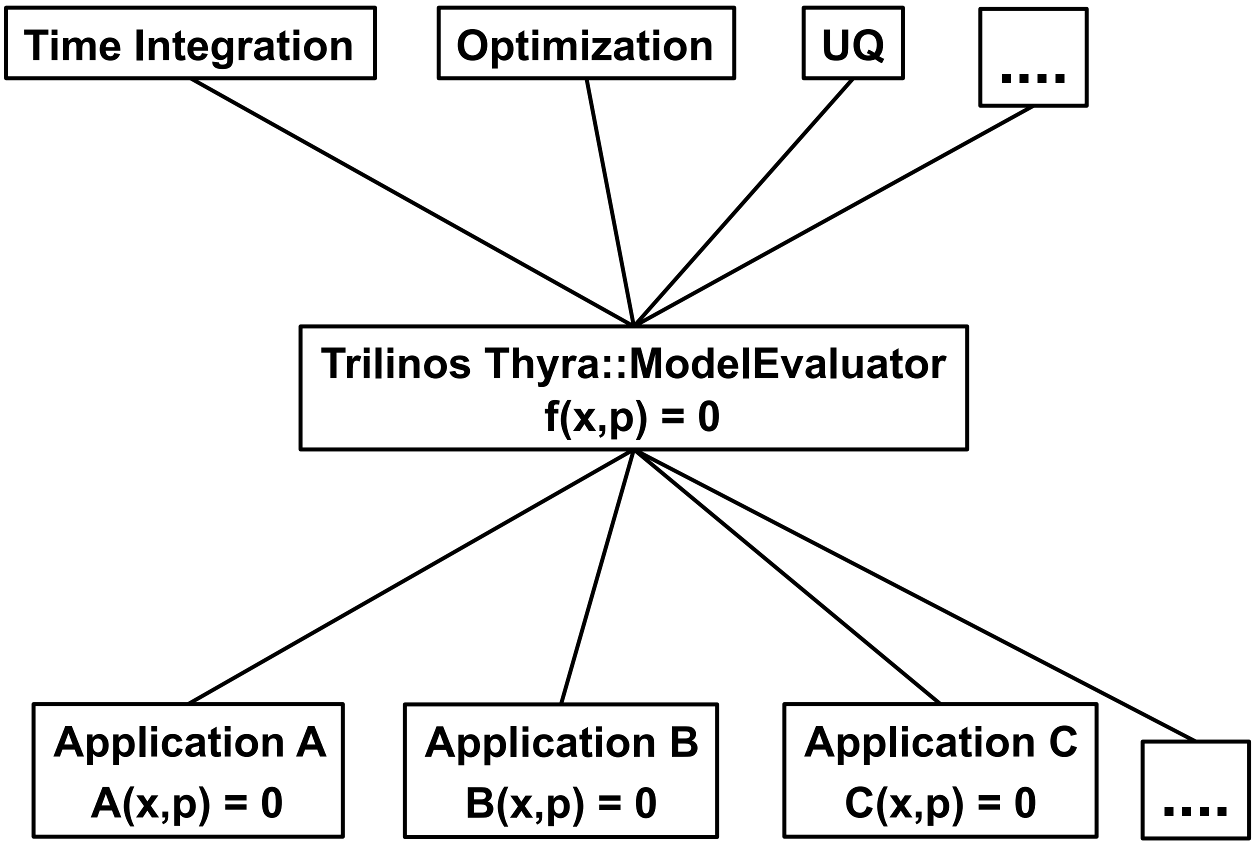
Interfacing Abstract Numerical Algorithms (ANA) To Applications



Issues:

- costly implementation
- duplication of effort
- difficult to maintain

Interfacing Abstract Numerical Algorithms (ANA) To Applications



- **Input requirements:**
  - State x
  - Parameters p
- **Output options:**
  - Residual f
  - Jacobian df/dx
  - Adjoint df/dx^T
  - Parameter derivs df/dp
  - Observation funcs g
  - …
- **Decorators:**
  - SG residuals/Jacobians
  - State elimination
  - Reduced sensitivities
  - …

Trilinos

http://trilinos.sandia.gov/

Sandia National Laboratories